Introducción a Ruby

Módulos

Overview

- Modulos
  - Utilizados como Namespaces
  - Utilizados como Mixins
- Utilización de los modulos built-in de ruby, especialmente Enumerable.
- **require_relative**

Módulos / Modules

- Es un **contenedor** de clases, métodos y constantes
  - o de otros **módulos**
- Es como una **clase** pero no puede ser **instanciada**.
- Las **clases** heredan de **Module** y agrega **new**

Módulos

- Sirve para dos propósitos:
  - Namespace
  - Mix-in

Módulos como namespace

```ruby
module Sports
  class Match
    attr_accessor :score
  end
end

module Patterns
  class Match
    attr_accessor :complete
  end
end

match1 = Sports::Match.new
match1.score = 45; puts match1.score # => 45

match2 = Patterns::Match.new
match2.complete = true; puts match2.complete # => true
```

Notar el uso del operador ::

Módulos como Mix-in

- En orientación a objetos existe lo que llamamos interfaces.
  - Son como contratos que definen lo que la clase puede hacer.
- Los mixins proveen una forma de compartir (o mezclar) código entre múltiples clases.

Se pueden incluir los módulos pre-construidos de ruby como Enumerable que puede hacer muchos trabajos importantes!

Módulos como Mix-in

```ruby
module SayMyName
  attr_accessor :name
  def print_name
    puts "Name: #{@name}"
  end
end

class Person
  include SayMyName
end
class Company
  include SayMyName
end
```

```ruby
person = Person.new
person.name = "Joe"
person.print_name
# => Name: Joe
company = Company.new
company.name = "Google & Microsoft LLC"
company.print_name
# => Name: Google & Microsoft LLC
```

Módulo Enumerable

- `map, select, reject, detect` etc.
- Es utilizado por la clase Array y muchas otras clases.
- Se puede incluir en una clase propia (nuestra).
  - Lo único que hay que hacer es proveer una implementación del método each.

Al incluir Enumerable e implementar el método each todas las demás funcionalidades están mágicamente disponibles.

include Enumerable

Ejercicio (practica-4/team_players.rb)

- Crear una clase **Player**:
  - Tiene los atributos: name, age, skill_level
  - Tiene un constructor que recibe los tres atributos
- Crear una clase **Team**:
  - Puede contener a varios jugadores.
  - Debe tener un constructor que reciba como splat a los jugadores.
- Crear un Team con 5 jugadores. (como está en **practica-4/team_players.rb**)
- Seleccionar únicamente los jugadores entre 14 y 20 años y no incluir a ningún jugador con skill-level menor a 4.5

Para incluir archivos externos

- Las clases se pueden programar en distintos archivos y pueden ser visibles para otras clases utilizando el comando
  - require_relative “….path”

```
require_relative 'player'
require_relative 'team'
```

Si la clase Player estuviera definida en un archivo player.rb
Y la clase Team estuviera definida en un archivo team.rb
Se podrían utilizar desde otro archivo utilizando el require_relative.

Entonces...

- Los módulos nos permiten hacer un mix-in de código útil dentro de otras clases.
- require_relative es útil para incluir otros archivos ruby relativos al actual código ruby.